CSC 584/484 Spring 25 Homework 1: Movement

Due: 01/29/2025 by the start of class

Overview

Your task for this assignment is to setup your development environment and explore basic movement of a sprite on the screen. Using *C++* and `SFML`, you will create a workflow that enables you to develop in C++ and ensure your code will be functional in the environment grading will take place. You will submit your code and a brief README only for this assignment.

This assignment is worth 25% of your homework grade.

This is an ***individual assignment***, you are to work alone. As always, you are expected to abide by the University's Academic Integrity Policy (`http://policies.ncsu.edu/policy/pol-11-35-01`), which includes providing appropriate attribution for all external sources of information consulted while working on this assignment.

Development vs. Grading Environments

In general, you are free to develop your code using any tools and any environment you wish; however, with such a large course it will be crucial that we standardize on an environment to facilitate grading. Regardless of what you choose to use for developing your code, you will have to:

1. Submit a `Makefile` that will allow your code to be compiled in a standardized Ubuntu 20.04 environment (details below).
2. Ensure that the resulting binary executes without errors in that standard environment.

The grading environment will be Ubuntu 20.04, with very little in the way of additional libraries installed. *Note: the instructor may opt to add libraries for future assignments.* As one part of this assignment, you will have to setup a grading environment.

You may choose to develop your code in a different environment than you validate grading in. For example, macOS can easily install SFML via homebrew (even on Apple Silicon). You may choose to use a fully-featured code editor like VS Code. This is all completely up to you; however, keep in mind that ***it is your responsibility to validate that your code compiles and runs in the grading environment as specified in this assignment and the instructions provided for setting it up.***

Part 1: Setup Grading Environment

The standardized grading environment will be Ubuntu 20.04 Desktop with `libsfml-dev` and `build-essential` installed. We will use the `gcc` compiler, with C++17 as the standard.

- If you're using MacOS, I recommend VMWare Fusion Player, which is free to use for students. You can access Fusion by registering for a free VMWare CustomerConnect account. See `https://www.vmware.com/products/fusion.html` for details.

- If you're running Windows or Linux, I recommend using VMWare Workstation Player (see here: `https://www.vmware.com/products/workstation-player.html`). Workstation Player is free.

- If you are already running Ubuntu 20.04 on the machine you plan to use for this course, I strongly suggest you setup a VM to test with to ensure there are no accidental dependencies you aren't aware of. In that case, Ubuntu's Multipass tool is a good alternative to VMWare for quickly and easily setting up a CLI-only VM which you can ssh to (locally) and display windows in your host environment.

- If you are using a Mac computer with Apple Silicon, you will need to start by installing the Ubuntu Server image (`https://cdimage.ubuntu.com/releases/20.04.5/release/ubuntu-20.04.5-live-server-arm64.iso`) and then installing the desktop environment with `sudo apt update && sudo apt -y upgrade && sudo apt install ubuntu-desktop`.

When you have completed this part of the assignment, you should have a functioning Ubuntu 20.04 installation with minimal packages installed.

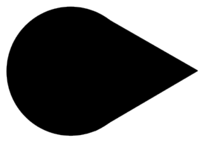
The packages that will be installed on top of the base system are: `build-essential` and `libsfml-dev`. If you depend on anything else, you run the risk of not getting credit for any implementation.

Once you have installed Ubuntu 20.04 Desktop (or server and then the `ubuntu-desktop` package), you should do any suggested software updates (`sudo apt update && sudo apt -y upgrade`). Then, to complete setup of the grading environment, run `sudo apt -y install build-essential libsfml-dev`. That will complete the installation of the grading environment.

Please note: you are free to use other libraries for developing your code, with the important caveat that they must be self-contained and not require the installation of any apt packages (as this would change the grading environment). If you do rely on third party libraries not included in the grading environment, make sure they are included in your submission to Moodle and your code compiles and runs properly in the standard grading environment using only the Makefile you provide.

Part 2: Moving a Sprite (20 points)

SFML has some very good content online for getting started. Please visit `https://www.sfml-dev.org/learn.php` for a plethora of tutorials to get familiar with the basics. For this part of the assignment, you overall task is to get a sprite drawn to the screen and moving around.



The SFML window should be VGA resolution (640x480). Moodle has two `.png` files available for download: "boid.png" and "boid-sm.png". The first being a higher-resolution 930x627 image and the second being a very low resolution 10x7 image. The smaller image is a good size for moving about the VGA resolution window size, but you should feel free to use your favorite tool to increase or decrease from those options.

Start by drawing that sprite to the screen, and then update it's position periodically (every 100 frames or so depending on the speed of your computer) in order to make it move left to right across the top of the screen. When it hits the right end of the screen, you should "teleport" it back to the left side.

For this part of the assignment, you will want to look at the `.move(...)` and `.setPosition(...)` functions in SFML's `sf::Sprite` class.

Part 3: Multiple Sprites (40 points) and More Complex Movement (40 points)

For this part of the assignment you will draw four sprites to the screen at the same time, and have them move in a coordinated pattern. The sprites should be offset from the edge of the screen sufficiently far so as to be completely visible.

1. Start with a single sprite near the top-left corner of the screen, then move it across to the top-right as you did for Part 1, but maintaining the proper offset to be completely visible.
2. When the first sprite reaches a point in the top-right corner similarly offset from the starting point, rotate it 90 degrees to point downwards and then add a second sprite at the starting location near the top-left. Make the first sprite move down the right side of the screen while the second sprite moves across the top. *Note: because the screen is not a square, you will have to move the first sprite a different number of pixels than the second in order to have them reach the next corners at the same time.*
3. When the first and second sprite reach the bottom-right and top-right corners respectively, rotate them each 90 degrees and add a third sprite at the top-left corner. Now move the first sprite along the bottom of the screen towards the bottom-left corner (maintaining the proper offset to be visible), the second sprite down the right side to the bottom-right corner (maintaining the proper offset), and the third sprite along the top towards the top-right corner (maintaining the proper offset). Again, you will have to adjust the number of pixels each sprite is moved in order to time them reaching the corners in sync.
4. When all three have reached the next corners, add a fourth sprite at the top-left and move all four sprites in coordination. The first sprite should end up back in the top-left corner, second in the bottom-left, third in the bottom-right, and fourth in the top-right.
5. When a sprite reaches the starting location in the top-left, it should be removed from the screen, but all the other sprites should continue.
6. When all four sprites have reached the top-left corner and been removed from the screen, begin again. This pattern should repeat until the window is closed.

For this part of the assignment, in addition to the functions used for Part 1, you will want to look into the `.rotate(...)` and `.setRotation(...)` functions and, possibly, the `sf::transform` class.

What to submit

By the start of class on 01/29/2025, please upload a .zip archive to Moodle. This archive should contain all of your code, a Makefile that will produce an executable in the grading environment (the sample Makefile from Moodle is fine if it works), and short README text file with instructions for how to compile and run your code to demonstrate both Part 2 and Part 3 of this assignment. You may elect to have your code for these parts in separate directories, just make sure the README file in the top-level directory explains how we can demonstrate both parts of your code.